# **Hackathon - Optisol Business Solutions**

**NAME: SAMRIDHA SRINIVASAN**

**ROLL NO: 71762132038**

**DEPARTMENT: M.Sc. Data Science**

## **AI-POWERED INSURANCE CLAIMS PROCESSING SYSTEM**

**PROBLEM STATEMENT:**

The insurance claims process is hampered by a high volume of diverse documents (PDFs, scanned images, Word docs) like claim forms, medical reports, and policy documents. Manual processing is slow and leads to delayed payouts and potential fraud.

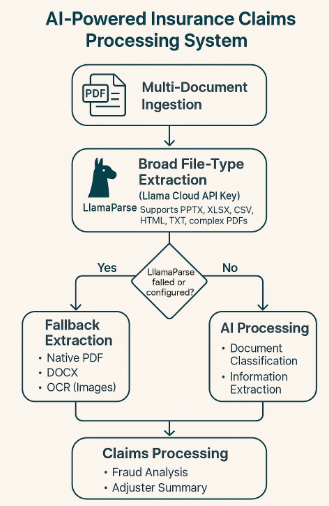
**TOOLS USED:**

* Streamlit – UI & workflow orchestration
* Python 3.9+
* OpenAI Python SDK – LLM calls
* LlamaParse (llama-parse, llama-index-core) – multi-format text extraction (PDF/DOCX/PPTX/XLSX/CSV/HTML/TXT)
* PyPDF2 – PDF fallback extractor
* python-docx – DOCX fallback extractor
* pytesseract + Tesseract OCR – image OCR (PNG/JPG/TIFF)
* Pillow (PIL) – image handling
* scikit-learn – TF-IDF + cosine similarity for exclusions check

**MODELS USED**

* OpenAI gpt-3.5-turbo – used for:
  + Document classification
  + Information extraction (JSON) per doc type
  + Adjuster summary generation
* Tesseract OCR – vision OCR model for text from images
* TF-IDF Vectorizer + Cosine Similarity (scikit-learn) – lightweight statistical model for exclusions matching
* Rule-based Fraud Scorer – heuristic scoring (no ML training)

**ARCHITECTURE:**



**IMPLEMENTATION:**

**Ingestion & Parsing**

1. For each file:
   * If image → OCRAdapter → raw\_text.
   * Else try LlamaParseAdapter → if empty/fails → PDF/DOCX fallback.
2. Build DocumentRecord{filename, mime\_type, raw\_text}; if empty, add warning.

**Classification (LLM)**

* Input: raw\_text[:N] (truncate for token safety, e.g., 2k chars).
* Output: doc\_type label.

**Extraction (LLM + JSON)**

* Choose prompt by doc\_type.
* Expect valid JSON; post-process: strip code fences, json.loads.
* Validate against schema; coerce types (dates, amounts), log missing fields.

**Exclusions Check (NLP)**

* Find diagnosis/ICD fields across documents (medical reports first).
* Compute TF-IDF cosine vs. exclusion phrases.
* If any match > threshold → matched=true.

**Fraud Scoring (Rules)**

* Load weights from config (sample below).
* Evaluate:
  + Amount: >100k → +25; >50k → +15
  + Missing required docs (CLAIM\_FORM, MEDICAL\_REPORT, BILL\_INVOICE): +10 each
  + Name inconsistency across docs (collect \*name\* keys): +25 if multiple unique
  + Same-day filing (incident\_date == claim\_date): +10
  + Low-quality docs (len(text) < 100): +5 each, cap +15
* Map score: 0–34 LOW, 35–59 MEDIUM, 60+ HIGH.

**Adjuster Summary (LLM)**

* Provide: claim metadata, condensed per-doc JSON, fraud analysis.
* Ask for sections (Key Details, Document Verification, Red Flags, Recommendation).
* Low temp (0.4) for consistency.

**Outputs**

* Show summary; enable TXT/JSON downloads.
* If HIGH RISK, optionally send Slack alert.

**CONCLUSION:**

This solution accelerates claim turnaround by automating ingestion and understanding of diverse documents, cutting manual effort and errors while delivering consistent, structured data that downstream systems can trust. It improves detection of high-risk cases with an explainable, rule-based fraud score, reducing leakage and enabling smarter human triage where it matters. Built-in validation against policy exclusions raises data quality and compliance confidence, and the standardized JSON outputs make integration straightforward with actuarial analytics, reporting, and core platforms. The approach scales to large volumes without sacrificing transparency, is configurable to your business thresholds, and ultimately elevates adjuster productivity and customer experience through faster, clearer, and more defensible decisions.